03

# Estandarizando retornos de API

## Transcripción

[00:00] Bien. Comencemos. Lo primero que tenemos que ver es que ya basado en lo que hemos hecho en el proyecto anterior, esta es una colección de request de Insomnia, que nos quedó de ahí.

[00:13] Insomnia, para los que están comenzando a ver este curso directamente, es el cliente, el aplicativo que simula ser un cliente para nuestro API, similar a Postman. Bueno, lo que tenemos hasta ahora, ¿qué es? Es un registro, un listado actualizar y un delete de médicos.

[00:30] Vamos a comenzar a ver algunas cosas que tienen en común, por ejemplo, el registrar médicos, vemos que la URL es exactamente la misma que el listado y la URL listado es la misma que actualizar. Y en el caso del delete, lo único que se le agrega es un parámetro más, en este caso, que es el id del médico. ¿Ahora por qué?

[00:50] Porque en Rest por más que tenga la misma URL, lo que manda es el verbo o método HTTP que tú quieras utilizar para interactuar con este recurso. Y sí, dije recursos porque en Rest la referencia que tenemos en los URLs son todas a nivel de recurso.

[01:12] Por ejemplo, aquí yo estoy diciendo en este dominio localhost:8080, mi servidor de mi máquina local, hace un post, o sea mándale un médico en este recurso médicos. Vamos a estar diciéndole: “Guárdame este recurso médico en los recursos de médico que tiene mi máquina actual.”

[01:34] En el caso de get, lo que estoy diciendo es “Dame todos tus recursos médicos”. Para actualizar es lo mismo y para el delete es aun más entendible porque básicamente borra de tu recursos médicos este recurso que tenga el id 7. Esas son unas particularidades de Rest.

[01:57] Primero, los métodos HTTP definen el tipo de operación que vas a realizar, segundo, la URL puede ser la misma, no hay ningún un problema si el método diferente. Y tercero tienes que referirte a recursos cuando hablas de rest services. ¿Qué más tenemos aquí? Algo muy curioso.

[02:17] Si vemos aquí en la mano derecha, aquí en verde, tenemos un código 200 que dice es el mismo para el update, el mismo para el listado y el mismo para el registro. ¿Qué significa el 200? 200 te dice la operación fue exitosa. Pero hay diferentes tipos de retorno que tenemos que aprender a hacer con Rest. ¿Por qué?

[02:40] Porque si bien yo sé por ejemplo que cuando yo guardo este médico, me da un 200 okay, yo necesito en realidad por buena práctica retornarle algo a mi cliente para que sepa que la operación fue exitosa, por ejemplo, hay un código http específico para cuando creas objetos que es el 202.

[03:00] Hay un código http específico para cuando realizas una operación y no retornas nada, que es el 204, es el no content. Entonces ese tipo de operaciones no necesita ser 200 necesariamente. Quizás el actualizar sí está okay, retornar un 200, pero para un delete, lo más apropiado sería retornar un 204.

[03:25] Significa que la operación fue exitosa pero no hay contenido que retornar porque hiciste un delete. Entonces esta es hora de ir al código, vamos a poner manos a la obra para recordar lo que vimos y por ejemplo aquí está nuestro método delete. Vemos que está nuestro id, actualizar, es lo mismo que hemos hecho en el curso pasado, no se preocupen.

[03:49] Y lo que yo voy a hacer ahora es actualizar este método para que me retorne algo, porque yo deseo tratar este código de HTTP. Yo quiero retornar

algo diferente, ¿pero cómo lo hago? Para eso Spring ya nos ayuda a retornar códigos ya más personalizados para que nuestra llamadas sean más personalizadas.

[04:08] Primero que todo tengo que cambiar el void, porque si bien el void, para los que son más puristas en Java, si bien el void retorna, no hay ningún problema, tú puedes retornar, es indiferente, no retorna en realidad nada al cliente, no retorna nada de valor en sí, simplemente retorna porque sale del método y listo.

[04:33] Pero en este caso yo quiero retornar un código HTTP, que es el 204. Para eso voy a usar la clase ResponseEntity de Spring. Y ahora sí me está dando un error de compilación. ¿Por qué? Porque si yo estoy declarando mi método ResponseEntity yo tengo que retornar un ResponseEntity.

[04:52] Y entonces aquí le voy a decir ResponseEntity. y ResponseEntity me da una serie de métodos estáticos. Vamos a darles un vistazo. Tenemos ok, noContent, accepted, badRequest, created, internalServerError, notFoud. ¿Se les hace familiar?

[05:10] Exacto. Son los códigos HTTP, los códigos de retorno HTTP que necesitamos. Por ejemplo, si yo le doy aquí un noContent internamente Spring va a mapear la respuesta que yo quiero y debería retornar un 204 a mi cliente. Vemos que no está compilando. ¿Esto es por qué?

[05:33] Porque para que retorne necesito darle un build, porque este método estático lo que va a generar internamente, va a setear el código de HTTP, pero necesito convertirlo con build a un ResponseEntity. Esto debería ser todo por ahora. Entonces voy a ejecutar mi servidor. Voy a ascenderlo y voy a juntar algún request para eliminar algún médico, vamos a ver qué médicos tenemos primero.

[06:03] Regreso a Insomnia, voy listar mis médicos. Quiero ver cuáles tengo. Tengo id 8 y 9. Entonces aquí en el delete le voy a decir: “Bórrame el médico

con el id 8". Le voy a dar send y ahora sí fíjese: 204 No Content. Y bueno, y eso mismo nos facilita una explicación acerca de qué es lo que significa este método.

[06:31] Ya de esta forma entonces ya estamos customizando, personalizando más nuestras llamadas HTTP. Vamos a ver este método de aquí arriba, el de actualizar médico. De igual forma yo deseo retornar algo, deseo retomar 200 pero por buena práctica de HTTP, de Rest, yo necesito retornar el objeto que acabo de actualizar.

[06:51] ¿Esto qué significa? Que este médico que yo actualizado, este de aquí, yo tengo que retornar sus datos, ¿cómo voy a hacer eso? Bueno, yo ya tengo mi médico acá, voy a actualizar los datos al médico, como es un transactional lo que él va a hacer es convertirlo en la base de datos, todo bien hasta ahí, y ahora yo necesito retornar es un responseEntity.

[07:16] Entonces, ResponseEntity, y vemos que ahora necesito retornar, voy a dar un return ResponseEntity.ok, porque yo he desarrollado un estado 200 que es ok, y aquí le puedo mandar médico, y como esto debe ser suficiente. ¿Pero cuál es el problema? Yo ya les expliqué que no es bueno retornar directamente la entidad de la base de datos.

[07:43] Recuerdan eso. Y este médico es la entidad de mi base de datos. ¿Qué es lo que yo tengo que hacer para evitar eso? Exacto, crear un DTO, o en todo caso un Java record aquí que haga el papel de DTO para interactuar con mi capa cliente. Aquí yo ya tengo creado un DTO llamado DatosRespuestaMédico, que lo tengo ya listo para evitar crearlo desde cero aquí otra vez.

[08:08] Y los parámetros que tiene son los mismos que tiene el médico: id, nombre, e-mail, teléfono, etcétera, y aquí tenemos lo mismo, es tal cual el mismo médico. Entonces con eso, ya tengo mi DTO para retornar. Aquí, ¿qué tengo que hacer ahora? Darle un new DatosRespuestaMedico y aquí voy a ir seteando uno a uno los valores que debería tener mi médico para retornar.

[08:43] Para fines de hacer este video un poco más rápido y no tener que copiar todo, yo tengo aquí ya el código listo para retornar el médico. Darle un vistazo, es lo que ustedes ya conocen del curso anterior: DatosRespuestaMedico con id, el nombre, email y en la dirección, he obtenido la dirección, la calle y bueno, todos los parámetros que ustedes ya conocen, son familiares con esto.

[09:07] Entonces ahora el método compila bien, porque tengo mi médico, lo actualizo y después lo que retorno son dos datos de mi DTO de mi médico. Voy a guardar aquí, voy a esperar que compile mi código nuevamente, vemos que está un put. Y regresamos aquí. Voy a listar mis médicos nuevamente para saber cuál quiero actualizar.

[09:33] Este el único que me queda con id 9. Betty Fleck, por ejemplo, voy a actualizar los datos de Betty Fleck. Entonces, en actualizar médico le voy a decir que nombre va a ser "Betty Updated" por ejemplo, solo para diferenciarla muy bien. Y el id era el id número 9.

[09:56] Entonces voy a venir aquí, voy a decir 9, lo voy a enviar. Me da 200 OK, y observen ahora que sí estoy obteniendo como respuesta la información de Betty, Betty Updated, el correo que tiene Betty, todo lo demás que no fue actualizado, pero ya este método ya me está retornando primero el código HTTP, que fue exitoso y también ya me está retornando el body del nuevo objeto, que ya existe a nivel de base de datos.

[10:30] ¿Por qué? Porque ya no me retorna Betty Fleck, ahora me retorna Betty Updated, a ver, entonces esto es solo el inicio, porque aún tenemos dos métodos más que tenemos que actualizar, que son el listado de médicos y el registro de médicos, pero eso es tema del siguiente video. Nos vemos.